Übungen zum Kapitel 11

Zeitreihen

Erstellt und überarbeitet: armin.baenziger@zhaw.ch, 15. Mai 2020

In [1]:
```
%autosave 0
```

```
Autosave disabled
```

(A.1) Laden Sie NumPy, Pandas und Matplotlib.pyplot mit der üblichen Abkürzung.

In [2]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
```

(A.2) Führen Sie den Magic Command aus, so dass Matplotlib-Plots "inline" erscheinen.

In [3]:
```
%matplotlib inline
```

(B.1) Die csv-Datei `hspitr.csv` im Ordner `weitere_Daten` enthält die Indexstände vom Swiss Performance Index (SPI) und den Subindizes *Small Cap*, *Mid Cap* und *Large Cap*. Laden Sie die Daten der Datei `hspitr.csv` in das DataFrame `Kurse` . Die erste Spalte enthält das Datum. Legen Sie dieses als Index (`index_col=0`) im Format `datetime` (`parse_dates=True`) fest.

In [4]:
```
Kurse = pd.read_csv('../weitere_Daten/hspitr.csv',
                    sep=';', parse_dates=True, index_col=0)
Kurse.head()
```

Out[4]:

Date	SPI	SPI_Small_Cap	SPI_Mid_Cap	SPI_Large_Cap
2017-12-29	10751.51	27688.03	16559.27	9990.81
2017-12-28	10779.42	27663.05	16616.74	10015.73
2017-12-27	10803.98	27615.38	16639.16	10041.51
2017-12-22	10756.39	27530.06	16541.18	10000.14
2017-12-21	10780.76	27464.56	16529.44	10030.27

(B.2) Sortieren Sie das DataFrame `Kurse` nach dem Datum (Index) aufsteigend. Tipp: Methode `sort_index` mit `inplace=True` verwenden.

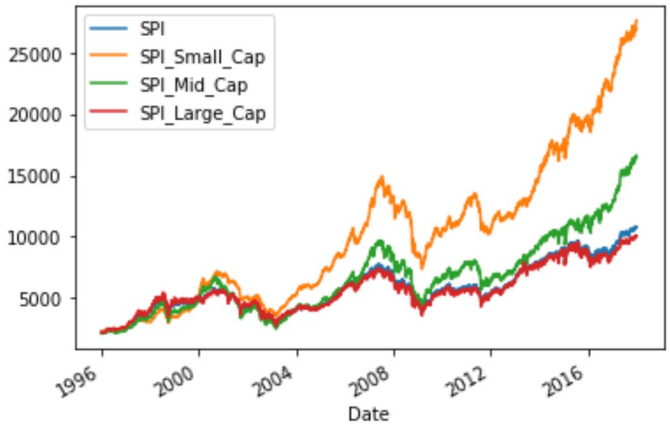
In [5]:
```
Kurse.sort_index(inplace=True)
Kurse.head()
```

Out[5]:

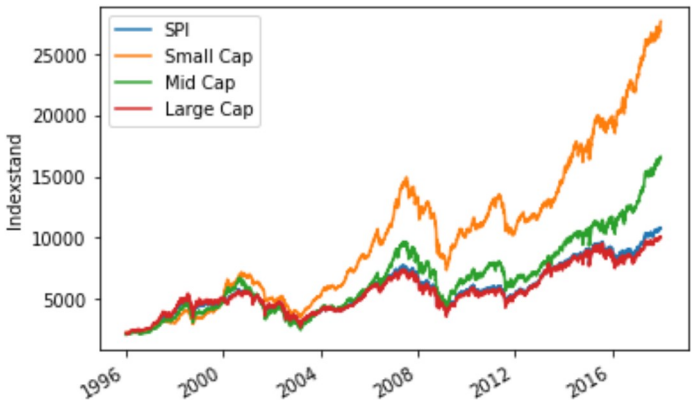
Date	SPI	SPI_Small_Cap	SPI_Mid_Cap	SPI_Large_Cap
1996-01-03	2172.31	2170.14	2040.95	2208.62
1996-01-04	2182.67	2184.72	2049.35	2219.26
1996-01-05	2175.09	2181.00	2054.88	2208.69
1996-01-08	2176.68	2194.96	2071.98	2206.46
1996-01-09	2174.13	2204.00	2077.84	2201.63

(B.3) Stellen Sie die Indexstände der vier Indexreihen in einem Liniendiagramm dar.

In [6]:
```
Kurse.plot();
```

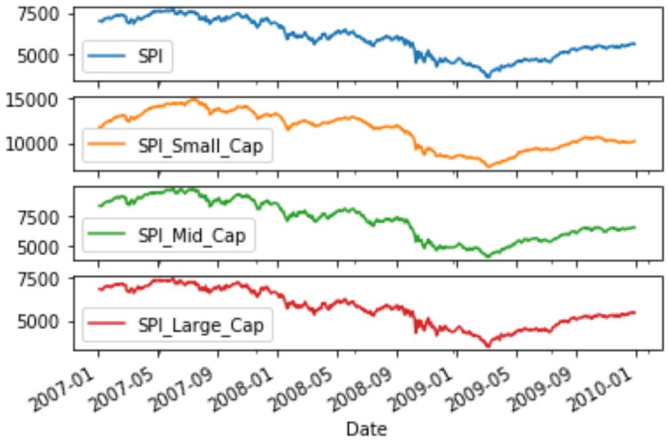


In [7]:
```
# Schöneres Diagramm:
Kurse.plot()
plt.legend(['SPI', 'Small Cap', 'Mid Cap', 'Large Cap'])
plt.xlabel('')
plt.ylabel('Indexstand');
```

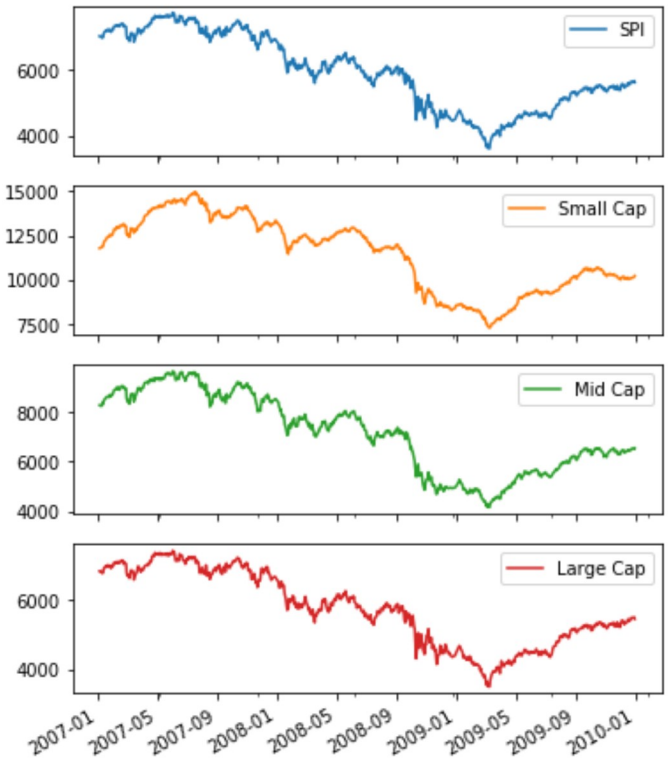


(B.4) Stellen Sie die Indexstände der vier Indexreihen *zwischen Anfnang 2007 und Ende 2009* in *separaten* Liniendiagrammen dar. Tipp: `plot` -Argument `subplots=True` verwenden.

In [8]:
```
Kurse['2007': '2009'].plot(subplots=True);
```



In [9]:
```
# Schöneres Diagramm
dic = {'SPI_Small_Cap': 'Small Cap',
       'SPI_Mid_Cap'  : 'Mid Cap',
       'SPI_Large_Cap': 'Large Cap'}
Kurse['2007': '2009'].rename(columns=dic).plot(subplots=True, figsize=(6,8))
plt.xlabel('');
```



(B.5) Stellen Sie die Entwicklung der vier Indexreihen zwischen Anfang 2007 und Ende 2009 in *einem* Liniendiagramm dar. Normieren Sie hierzu alle Indexreihen auf den Wert 100 zum Ausgangszeitpunkt. Tipp: Durch erste Zeile teilen und mal 100 rechnen.

```
In [10]: Kurse0709 = Kurse['2007': '2009']
         (Kurse0709/Kurse0709.iloc[0]*100).plot();
```

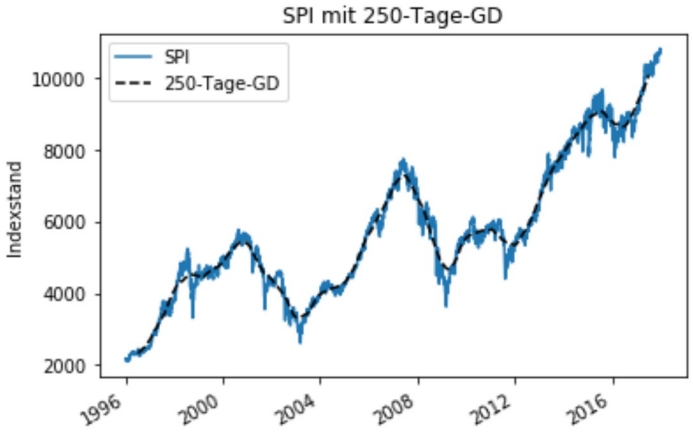
(B.6) Ermitteln Sie die Anzahl Handelstage in jedem Jahr im DataFrame `Kurse` . Tipp: `Kurse.index.year` gibt das Jahr zurück. Davon kann dann eine Häufigkeitstabelle erstellt werden.

```
In [11]: Kurse.index.year.value_counts().sort_index()
```

```
Out[11]: 1996    252
         1997    251
         1998    251
         1999    253
         2000    251
         2001    250
         2002    253
         2003    250
         2004    254
         2005    255
         2006    251
         2007    249
         2008    251
         2009    251
         2010    254
         2011    254
         2012    250
         2013    249
         2014    249
         2015    251
         2016    254
         2017    251
         Name: Date, dtype: int64
```

(B.7) Stellen Sie den Verlauf des SPI-Indexes in einem Liniendiagramm dar. Fügen Sie zudem einen einfachen (zentrierten) gleitenden Durchschnitt der Ordnung 250 dem Diagramm hinzu. (250 Handelstage entsprechen ungefähr einem Jahr, wie oben bestätigt wurde.)

In [12]:
```
Kurse.SPI.plot(title='SPI mit 250-Tage-GD')
Kurse.SPI.rolling(window=250,
                  center=True).mean().plot(style='k--')
plt.legend(['SPI', '250-Tage-GD'])
plt.xlabel('')
plt.ylabel('Indexstand');
```



(C.1) Bilden Sie aus dem DataFrame `Kurse` das DataFrame `Renditen`, welches die *stetigen* Tagesrenditen der vier Indizes *in Prozent* umfasst. Tipp: $r_{t,stetig} = \ln(K_t/K_{t-1}) \cdot 100$. Den natürlichen Logarithmus können Sie mit `np.log()` bilden.

In [13]:
```
Renditen = np.log(Kurse/Kurse.shift(1)).dropna()*100
Renditen.head()
```

Out[13]:

Date	SPI	SPI_Small_Cap	SPI_Mid_Cap	SPI_Large_Cap
1996-01-04	0.475778	0.669599	0.410728	0.480592
1996-01-05	-0.347886	-0.170419	0.269478	-0.477423
1996-01-08	0.073074	0.638034	0.828722	-0.101016
1996-01-09	-0.117220	0.411007	0.282422	-0.219143
1996-01-10	-1.155617	-1.121052	-0.438917	-1.302947

(C.2) Berechnen Sie die mittleren Renditen des SPI, *gruppiert nach Wochentag*. Tipp: Die Wochentage erhalten Sie mit `Renditen.index.weekday`, `Renditen.index.dayofweek` oder `Renditen.index.day_name()`.

In [14]:
```
Renditen.SPI.groupby(Renditen.index.day_name()).mean()
```

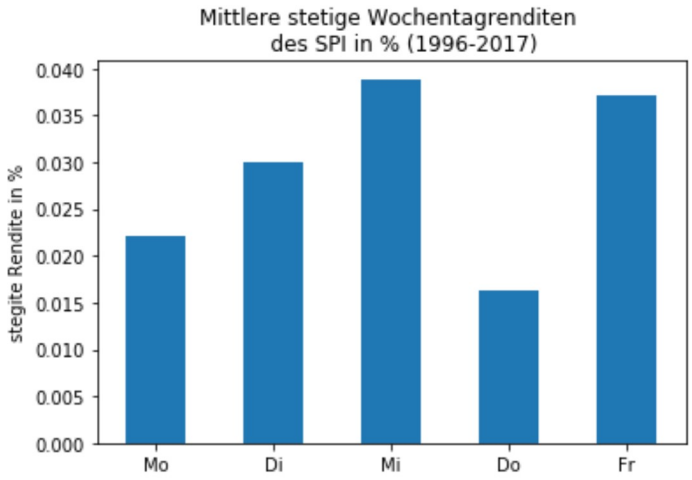
Out[14]:
```
Date
Friday       0.037103
Monday       0.022041
Thursday     0.016202
Tuesday      0.029969
Wednesday    0.038865
Name: SPI, dtype: float64
```

In [15]:
```
# oder sortiert nach Wochentag:
tabelle = Renditen.SPI.groupby(Renditen.index.dayofweek).mean()
# Etwas Ästhetik:
tabelle.rename({0: 'Mo', 1: 'Di', 2: 'Mi', 3: 'Do', 4:'Fr'},
               inplace=True)
tabelle
```

Out[15]:
```
Date
Mo    0.022041
Di    0.029969
Mi    0.038865
Do    0.016202
Fr    0.037103
Name: SPI, dtype: float64
```

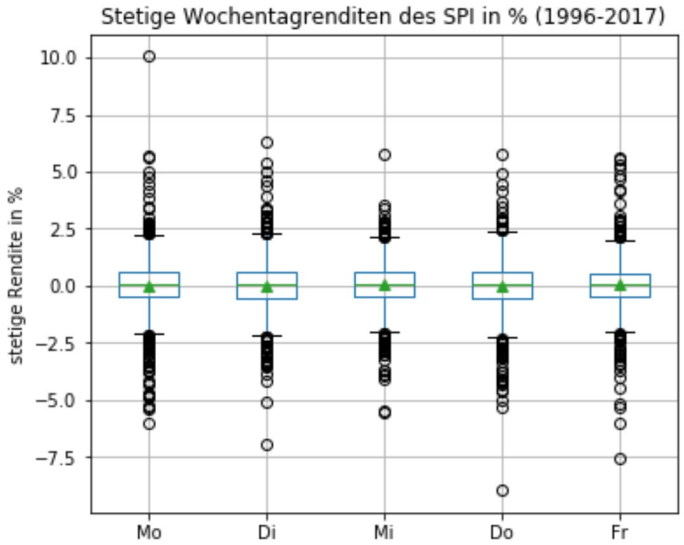
(C.3) Stellen Sie die mittleren stetigen Tagesrenditen des SPI nach Wochentag in einem Säulendiagramm dar.

In [16]:
```
tabelle.plot.bar(rot=0,
    title='''Mittlere stetige Wochentagrenditen
    des SPI in % (1996-2017)''')
# Hinweis: Mit ''' kann man über Zeilen hinweg schreiben.
plt.xlabel('')
plt.ylabel('stegite Rendite in %');
```



Hinweis: Statt lediglich die Mittelwerte mit Säulendiagrammen darzustellen, sollte man besser die *ganzen Verteilungen* mit Boxplots charakterisieren. Dadurch kann man erkennen, dass die Streuungen der Tagesrenditen sehr gross sind relativ zu den Mittelwerten.

In [17]:
```
wochentag = Renditen.index.dayofweek
df = Renditen.pivot_table(columns=wochentag,
                          index=Renditen.index,
                          values='SPI')
df.boxplot(showmeans=True, figsize=(6,5))
plt.title('Stetige Wochentagrenditen des SPI in % (1996-2017)')
plt.xticks(range(1,6), ['Mo', 'Di', 'Mi', 'Do', 'Fr'])
plt.ylabel('stetige Rendite in %');
```



(D.1) Bilden Sie aus dem DataFrame `Renditen` ein DataFrame `RenditenM` mit stetigen *Monatsrenditen*.

- Tipp: Methode `resample` und `sum` verwenden.
- Hinweis: Stetige Renditen sind additiv, das heisst, dass die Summe der Tagresrenditen eines Monats die entspprechende Monatsrendite ist.

In [18]:
```
RenditenM = Renditen.resample('M', kind='period').sum()
# Lösung ohne kind='period' ist auch ok.
RenditenM.head()
```

Out[18]:

Date	SPI	SPI_Small_Cap	SPI_Mid_Cap	SPI_Large_Cap
1996-01	-2.330485	1.452440	2.074897	-3.381693
1996-02	2.580137	2.088498	1.524028	2.819586
1996-03	6.563023	0.759884	3.477085	7.420033
1996-04	1.137367	3.845629	1.734168	0.908183
1996-05	-2.270421	0.229102	-0.875092	-2.659714

(D.2) Wenden Sie die Methode `describe` auf die Monatsrenditen an.

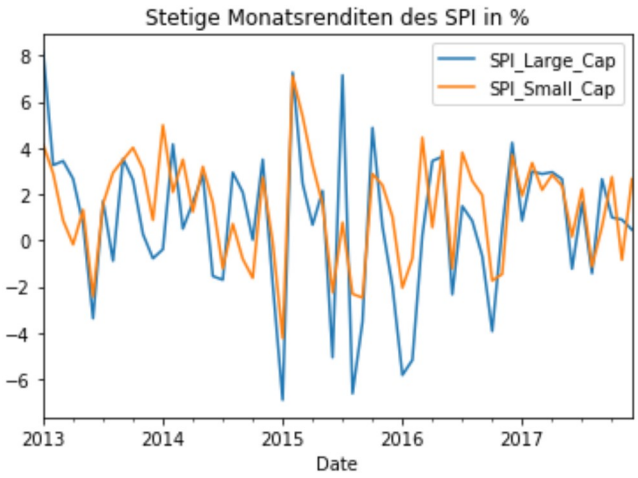
In [19]:
```
RenditenM.describe()
```

Out[19]:

	SPI	SPI_Small_Cap	SPI_Mid_Cap	SPI_Large_Cap
count	264.000000	264.000000	264.000000	264.000000
mean	0.605778	0.964473	0.793004	0.571704
std	4.355844	4.091986	4.995344	4.439146
min	-19.930830	-16.258567	-23.480701	-20.058015
25%	-1.335059	-1.005961	-0.941702	-1.546005
50%	1.407643	1.486486	1.510637	1.066656
75%	3.250346	3.280611	3.726847	3.259442
max	11.549106	10.997769	14.202401	11.885973

(D.3) Erstellen Sie ein Liniendiagramm der stetigen Monatsrenditen des SPI-Large-Cap und des SPI-Small-Cap (nur diese zwei Indizes) *ab dem Jahr 2013*.

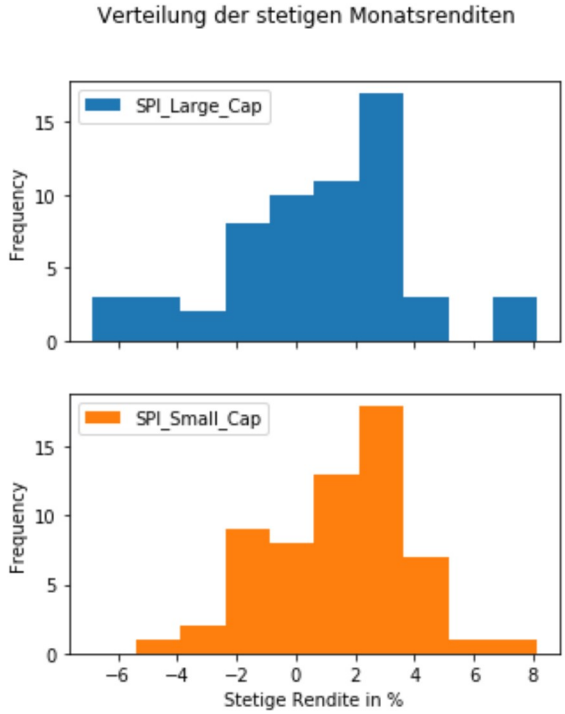
In [20]:
```
Plotdaten = RenditenM.loc['2013':, ['SPI_Large_Cap', 'SPI_Small_Cap']]
Plotdaten.plot(title='Stetige Monatsrenditen des SPI in %');
```



(D.4) Stellen Sie nun die stetigen Monatsrenditen des SPI-Large-Cap und des SPI-Small-Cap *ab dem Jahr 2013* in *separaten* Histogrammen dar.

Plotdaten.hist() geht auch (ohne plot vor hist und ohne subplots = True)

In [21]:
```
Plotdaten.plot.hist(subplots=True, figsize=(5,6),
                    title='Verteilung der stetigen Monatsrenditen')
plt.xlabel('Stetige Rendite in %');
```



(E.1) Berechnen Sie die Korrelationsmatrix der vier Indexreihen, einmal mit den Tages- und einmal mit den Monatsrenditen.

In [22]:
```
Renditen.corr()
```

Out[22]:

	SPI	SPI_Small_Cap	SPI_Mid_Cap	SPI_Large_Cap
SPI	1.000000	0.647529	0.853160	0.997021
SPI_Small_Cap	0.647529	1.000000	0.754967	0.610797
SPI_Mid_Cap	0.853160	0.754967	1.000000	0.811960
SPI_Large_Cap	0.997021	0.610797	0.811960	1.000000

In [23]:
```
RenditenM.corr()
```

Out[23]:

	SPI	SPI_Small_Cap	SPI_Mid_Cap	SPI_Large_Cap
SPI	1.000000	0.721666	0.857066	0.993748
SPI_Small_Cap	0.721666	1.000000	0.887371	0.654611
SPI_Mid_Cap	0.857066	0.887371	1.000000	0.795736
SPI_Large_Cap	0.993748	0.654611	0.795736	1.000000

(E.2) Berechnen Sie die (Auto-) Korrelation der stetigen SPI-Monatsrenditen mit den Vormonatsrenditen. Tipp: Die Vormonatsrendite erhält man mit `RenditenM.SPI.shift(1)` .

```
In [24]: RenditenM.SPI.corr(RenditenM.SPI.shift(1)).round(3)
```

```
Out[24]: 0.176
```

Die Monatsrenditen sind (in diesem Sample!) sehr schwach *positiv* autokorreliert. Man spricht in der Finance von einem "Momentumeffekt".

Ende der Übung